Amendments to the Claims

Kindly amend claims 1, 5-8, 11 & 16-18, and cancel claims 9, 10, 14 & 15 (without prejudice) as set forth below. All pending claims are reproduced below, with changes in the amended claims shown by underlining (for added matter) and strikethrough/double brackets (for deleted matter).

1. (Currently Amended) In a shared memory model system, a method whereby, in a state wherein a plurality of threads exist, a bit that represents a lock type and an identifier for a thread that has acquired a lock in accordance with a first lock type, or an identifier of a second lock type, are stored in a storage area that corresponds to an object and a lock on an object is thus managed, said method comprising:

determining, if a second thread attempts to acquire a lock on a specific object that ~~is held by~~ a first thread has accessed, whether a bit that represents said lock type on said specific object represents said first lock type;

setting a contention bit if said bit represents said first lock type;

determining, before said first thread unlocks said specific object, whether said bit that represents said lock type represents said first lock type;

responsive to the determining, storing in said storage area a special identifier that differs from the identifiers for said plurality of threads, the special identifier indicating that a two-stage unlocking is to be performed, wherein only one synchronization command is needed for unlocking when no contention exists;

issuing a synchronization command for said memory system;

storing in said storage area data indicating [[the]] absence of a thread that holds said lock on said specific object;

determining whether said contention bit has been set if said bit that represents said lock type represents said first lock type; and

terminating an unlocking process if said contention bit has not been set without any other process being performed.

2. (Original) The lock management method according to claim 1, further comprising:

shifting said first thread, when said contention bit has been set, to an exclusive control state for a mechanism that enables the exclusive control of the accessing of said object, and a thread waiting operation and the transmission to a waiting thread of a notification, both of which are to be performed when a predetermined condition has been established;

permitting said first thread to transmit said notification to said waiting thread;

setting said second thread in the busy waiting state, when said predetermined condition has not been established and when said special identifier has been stored, until a thread that holds said lock on said specific object is no longer present and until said bit that represents said lock type represents said first lock type; and

permitting said first thread to exit said exclusive control state.

3. (Original) The lock management method according to claim 1, wherein said first lock type is a lock method whereby to manage a lock state an identifier for a thread that has locked an object is stored in correlation with said object.

4. (Original) The lock management method according to claim 1, wherein said second lock type is a lock method whereby a queue is employed to manage a thread that has locked an access to an object.

5. (Currently Amended) In a shared memory model system, [[an]] a computer-implemented apparatus where, in a state wherein a plurality of threads exist, a bit that represents a lock type and an identifier for a thread that has acquired a lock in accordance with a first lock type, or an identifier of a second lock type, are stored in a storage area that corresponds to an object and a lock on an object is thus managed, said computer-implemented apparatus comprising:

means for determining, if a second thread attempts to acquire a lock on a specific object that ~~is held by~~ a first thread has accessed, whether a bit that represents said lock type on said specific object represents said first lock type;

means for setting a contention bit if said bit represents said first lock type;

means for determining, before said first thread unlocks said specific object, whether said bit that represents said lock type represents said first lock type;

responsive to the means for determining, means for storing in said storage area a special identifier that differs from the identifiers for said plurality of threads, the special identifier indicating that a two-stage unlocking is to be performed, wherein only one synchronization command is needed for unlocking when no contention exists;

means for issuing a synchronization command for said storage area;

means for storing in said storage area data indicating [[the]] absence of a thread that maintains said lock on said specific object;

means for determining whether said contention bit has been set if said bit that represents said lock type represents said first lock type; and

means for terminating an unlocking process if said contention bit has not been set without any other process being performed.

6. (Currently Amended) The computer-implemented lock management apparatus according to claim 5, further comprising:

means for shifting said first thread, when said contention bit has been set, to an exclusive control state for a mechanism that enables the exclusive control of the accessing of said object, and a thread waiting operation and the transmission to a waiting thread of a notification, both of which are to be performed when a predetermined condition has bee established;

means for permitting said first thread to transmit said notification to said waiting thread;

means for setting said second thread in the busy waiting state, when said predetermined condition has not been established and when said special identifier has been stored, until a thread that maintains said lock on said specific object is no longer present and until said bit that represents said lock type represents said first lock type; and

means for permitting said first thread to exit said exclusive control state.

7. (Currently Amended) The computer-implemented lock management apparatus according to claim 5, wherein said first lock type is a lock method whereby to manage a lock state an identifier for a thread that has locked an object is stored in correlation with said object.

8. (Currently Amended) The computer-implemented lock management apparatus according to claim 5, wherein said second lock type is a lock method whereby a queue is employed to manage a thread that has locked an access to an object.

9. (Canceled).

10. (Canceled).

11. (Currently Amended) In a shared memory model system, a method whereby, in a state wherein a plurality of threads exist, a bit that represents a lock is stored in a storage area that corresponds to an object, and a queue of threads that are waiting for access to said object is stored to manage a lock on an object, said method comprising:

determining, when a second thread attempts to acquire a lock on a specific object that a first thread has accessed ~~locked~~, whether a bit that represents said lock on said object represents the locked state;

changing, when said bit represents said locked state, data for the number ~~of queues~~ of threads in said queue that are waiting for [[can]] access to said specific object and storing the updated data, and thereafter issuing a synchronization command for said storage area;

storing said second thread in a queue, and shifting said second thread to a control state for a mechanism that performs a waiting operation, for accessing said specific object, and a recovery operation by transmitting a notification;

storing in said storage area, before said first thread unlocks said object, said bit that represents said locked state and an identifier that is not related to the representation of said locked state or an unlocked state, the identifier indicating that a two-stage unlocking is to be performed, wherein only one synchronization command is needed for unlocking when no contention exists;

issuing a synchronization command for said storage area;

storing, in said storage area, data that does not represent said lock on said specific object;

determining whether a thread that is stored in a queue is present;

shifting, when a thread that is stored in a queue is present, said first thread to a notification state wherein said transmission is initiated for issuing a notification to said thread that is waiting; and

permitting said first thread to exit said notification state.

12. (Original) The lock management method according to claim 11, further comprising:

increasing, when said bit that represents said locked state is set, the number of queues of threads that can access said specific object and storing the updated number, and determining whether said bit that represents said lock on said specific object represents said locked state; and

reducing, when said bit that represents said locked state is not set, the number of said queues of said threads that access said specific object and storing the updated number, and terminating a locking process without any other process being performed.

13. (Original) The lock management method according to claim 12, further comprising:

permitting said second thread, when said bit that represents said locked state is set and when an identifier that is not related to the representation of said locked state or said unlocked state is stored in said storage area, to remain in a busy waiting state until a thread that maintains said lock on said object is no longer present and said bit that represents said locked state is changed to represent said unlocked state.

14. (Canceled).

15. (Canceled).

16. (Currently Amended) In a shared memory model system, [[an]] a computer-implemented apparatus where, in a state wherein a plurality of threads exist, a bit that represents a lock is stored in a storage area that corresponds to an object, and a queue of threads that are waiting for access to said object is stored to manage a lock on an object, said computer-implemented apparatus comprising:

means for determining, when a second thread attempts to acquire a lock on a specific object that a first thread has accessed ~~locked~~, whether a bit that represents said lock on said object represents the locked state;

means for changing, when said bit represents said locked state, data for the number ~~of queues~~ of threads in said queue that are waiting for [[can]] access to said specific object and storing the updated data, and thereafter issuing a synchronization command for said storage area;

means for storing said second thread in a queue, and shifting said second thread to a control state for a mechanism that performs a waiting operation, for accessing said specific object, and a recovery operation by transmitting a notification;

means for storing in said storage area, before said first thread unlocks said object, said bit that represents said locked state and an identifier that is not related to the representation of said locked state or an unlocked state, the special identifier indicating that a two-stage unlocking is to be performed, wherein only one synchronization command is needed for unlocking when no contention exists;

means for issuing a synchronization command for said storage area;

means for storing, in said storage area, data that does not represent said lock on said specific object;

means for determining whether a thread that is stored in a queue is present;

means for shifting, when a thread that is stored in a queue is present, said first thread to a notification state wherein said transmission is initiated for issuing a notification to said thread that is waiting; and

means for permitting said first thread to exit said notification state.

17. (Currently Amended) The computer-implemented lock management apparatus according to claim 16, further comprising:

means for increasing, when said bit that represents said locked state is set, the number of queues of threads that can access said specific object and storing the updated number, and determining whether said bit that represents said lock on said specific object represents said locked state; and

means for reducing, when said bit that represents said locked state is not set, the number of said queues of said threads that access said specific object and storing the updated number, and terminating a locking process without any other process being performed.

18. (Currently Amended) The computer-implemented lock management apparatus according to claim 17, further comprising:

means for permitting said second thread, when said bit that represents said locked state is set and when an identifier that is not related to the representation of said locked state or said unlocked state is stored in said storage area, to remain in a busy waiting state until a thread that maintains said lock on said object is no longer present and said bit that represents said locked state is changed to represent said unlocked state.

* * * * *